

Chicken Object Detection and Tracking to Monitor Animal Welfare

Summer 2022 Practicum

 - Video Team 1 (V1) Final Report

Alex Dreo (adreo3), Ryan Elson (relson6), Evan Reed (ereed30)

Introduction	4
Company Overview	4
Problem Statement/Project Purpose	4
Methods and Experiment Design	5
Exploratory Data Analysis	5
Labeling	8
Base Model Selection	9
Base Feature Extraction	9
Pre Training Image Dataset	10
Image downsampling size	11
Performance Metrics	12
Precision and Recall	12
Intersection over Union (IoU)	13
Loss Metrics	13
Selected model	13
Model Creation Process	13
Output Analysis	14
Predicted Class Filtering	14
Model Tuning	15
Model Shortcomings	16
Overexposure	16
Underexposed and Distorted regions	17
Tight groupings of chickens	17
Labeling Differences	17
Model Application	18
SORT Object Tracking	18
SORT Background	18
SORT Implementation	19
SORT Results	21
Heatmap	26
Heatmap Tool Details	27
Considerations	29
Challenges	30
Conclusion	30

Next Steps	31
Normalize Brightness and Distortion	31
Label Additional Types of Objects or Behaviors	31
Increase The Video Frame rate	31
Augment the Training Data	32
References	33

Team V1 [REDACTED]: Final Report

1 INTRODUCTION

1.1 Company Overview

[REDACTED] Working as part of the Foundation for Food and Agricultural Research's (FFAR) and McDonald's SMART Broiler program, their services actively monitor poultry houses to detect anomalies and improve the welfare of the broiler chickens in these facilities.

1.2 Problem Statement/Project Purpose

In addition to sound data, [REDACTED] captures video feeds from inside the poultry houses. The purpose of this project is to utilize the videos to expand the scope of the anomaly detection and further increase animal welfare and supply chain efficiency. As evidenced by animal welfare laws in the EU and countries around the world, animal welfare is of paramount importance. And according to FFAR, "Healthy chickens have the highest feed efficiency, are more productive and profitable and are better for the environment (<https://foundationfar.org/>)."

Applying computer vision machine learning techniques, we create tools to analyze the video feeds of a poultry house to identify specific behaviors (of individuals or the entire flock).

Some examples of using video to identify behaviors that may indicate farmer intervention is required are:

- Animal proximity (Too cold if huddled)
- Animal eating and drinking behavior
- Animal interactions (e.g., fighting or play behavior)
- Animal movement (e.g., mass movements)

As video data requires more storage and is more expensive to process, the output from the vision models will need to be able to derive insights that could not be produced by the anomaly detection models built only using sound.

We chose to focus on detecting animal movement, specifically mass movements. Mass movements can indicate the chickens are frightened, scared, or otherwise disturbed in a way that is detrimental to their well-being. This may be due to a

person in the house, predator noises outside the house, or any number of other factors.

Detecting mass movements is beneficial because the farmer can attempt to mitigate or eliminate the root cause. This decreases stress and improves animal welfare, but minimizing mass movements is also beneficial financially to the farmer and consumer.

Consider that the more the chickens move (as in mass movement events), the more calories they burn. This negatively impacts the feed-to-meat gain (i.e., food conversion ratio) and means that the chickens need to eat more food to produce the same amount of meat for the consumer. This increases production costs, consumer costs, and contributes to climate change.


In 2020, the feed-to-meat gain was 1.79 (i.e., 1.79 pounds of feed consumed to produce 1 pound of broiler chicken weight) per the National Chicken Council (NCC), and since the average American ate 98 pounds of chicken (NCC's 2020 annual report) and 59 Billion pounds of chicken were produced in the USA in 2020 (USDA Poultry 2020 Summary), even small improvements have outsized impacts.

Healthy chickens keep costs down, help ensure supply can keep up with demand, and helps minimize climate change impacts.

2 METHODS AND EXPERIMENT DESIGN

2.1 Exploratory Data Analysis

A key component of our exploratory data analysis was to understand basic chicken behaviors that we could analyze. Broiler chickens' average market age is approximately six weeks (47 days according to the National Chicken Council), and their behaviors change drastically over that time.

Our first step was learning from the expertise of  to gain an initial understanding of animal behavior and welfare within the poultry houses.

Next, we watched videos from different houses over different weeks of the chickens' lives. This provided a basic visual perception of how animal behaviors change over time, while also calling attention to issues we might encounter during different weeks of their life.

Available videos from [REDACTED] covered the six weeks of the birds' lives spent in the houses and there were often two video channels to choose from for each of several houses. Some older video clips were also made available.

When chicks are first introduced to the house, the overhead lights are kept on during the day which is beneficial for building object detection models as it minimizes harsh lighting. At this age, the birds are smaller which creates more open ground space in the video frames. Taking this into consideration, we speculated that detection might be easier as a result. However, at this age, the feeder trays are placed on the ground and the birds are not yet using the standard feeders. This meant we would not be able to analyze standard feeding behaviors the birds exhibit over the majority of their lives. According to [REDACTED] at this age birds may be more likely to huddle but may also move around more so it might be challenging to detect anomalous movement events reliably.

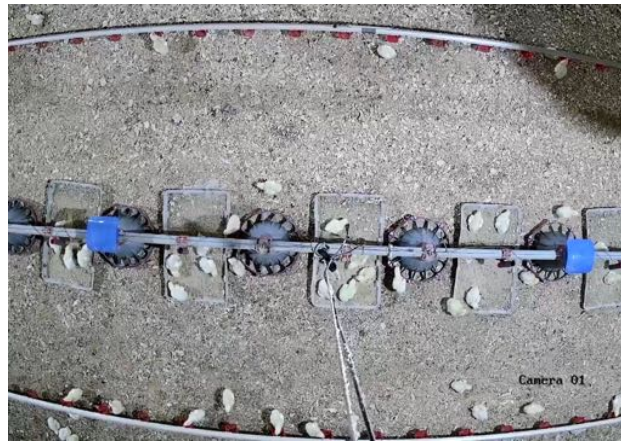


Figure 1: Young chickens after introduction to the house

After a few days, the chicks start to get bigger and the overhead lighting is no longer used. These lights are replaced with point lighting along the feeder line. Feed trays start to be removed and the chickens start using the standard feeders. The harsh, reflective lighting from these sources creates challenges when labeling and performing object detection, but the chickens are still small enough that there is a good deal of open ground.



Figure 2: Overhead lighting is no longer used. Few feed trays in service.

Older birds are much larger than younger birds, so we decided to avoid this age due to potential problems distinguishing between birds that were close to each other and the lack of open ground space. There was also less opportunity to detect different behaviors that are more common in younger birds.



Figure 3: Densely packed larger birds

Ultimately, we opted to focus our model on chickens between the ages of 1-2 weeks, as we felt it was best suited for a computer vision model. We suspected that any detected mass movement events may be more reliable than when the birds are younger as the size at this age allows for open space for movement tracking. The birds are young enough that we might observe instances of anomalous behavior, and at this age they are beginning to use the standard feeders, which is also a point of interest.

2.2 Labeling

After defining the scope of the model during exploratory data analysis, the training data was generated to leverage the supervised learning models for object detection. This was completed using Amazon Web Services Sagemaker Ground Truth, a tool that allows users to complete labeling jobs for multiple types of input data. Our analysis consisted of a random subset of frames from a video of the desired age range.

The specific labeling practice used in this project was bounding boxes. This requires that each chicken within the frame be identified by drawing a box around it. The boxes indicate that the subsets of pixels inside represent the image of a chicken indicating that other sets of pixels that exhibit similar characteristics might be a chicken as well.

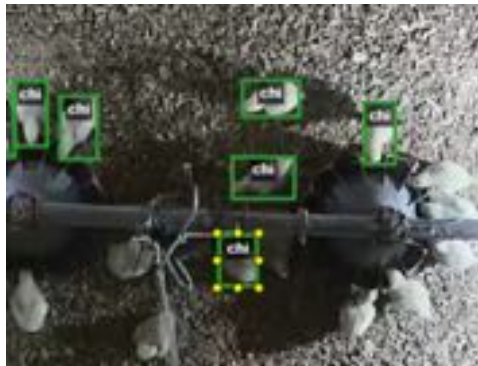


Figure 4: Ground Truth labeling

The rationale to apply the bounding box labeling technique was to gather new insights over previous analysis done with semantic segmentation. There was some initial concern with using bounding boxes, as previous attempts yielded poor results. However, the previous implementations used bounding boxes within small subsections of the frame. Labeling chickens within these smaller subsections resulted in many partial chickens (only a wing or head visible) being labeled in the same class, which resulted in the model identifying a single chicken in the frame as multiple objects. As the images being labeled in this project were not being divided into smaller images this was less of a concern, but we were still careful to label what we thought to be fully visible chickens.

After labeling 21 full sized images, containing over 6000 instances of chickens, the next step was to convert the labels as Amazon Ground Truth's object detection annotation files are not in a standard format. To convert the Ground Truth output manifest file into a usable format, we followed an AWS article ("Streamlining Data Labeling for YOLO Object Detection in Amazon SageMaker Ground Truth"), modifying code as necessary to create CSV files with our labels. We

wrote code to further process these CSV files into the JSON file format that we needed to train our model through Sagemaker Jumpstart. Once converted, a dataset was available to begin training models.

2.3 Base Model Selection

Several neural network architectures, pre-training image sets, and neural network resolutions were explored to achieve the goal of detecting and tracking the animals. The decision process is documented below.

Base Feature Extraction

There were several base feature extraction models considered, including YOLOv5, resnet50, and MxNet. These models had different formats required in terms of the labeling data. While we transformed our label data into the required format for each of these models, it was decided not to train a YOLO model, as Sagemaker Jumpstart is not capable of custom training this type of model.

Of the remaining two feature extraction models, we expected resnet to outperform vgg, as it did in a series of benchmarking tests (“Detection — GluonCV” n.d.). From our results however, they performed very similarly, with resnet actually producing more false positives, as can be seen in figure 5 and 7 below.



Figure 5: *jumpstart-ftc-mx-od-ssd-512-resnet50-v1-coco-adreo*: Predicted chicken objects using **resnet** architecture



Figure 6: Example chicken false positive

Pre Training Image Dataset

Training a neural network from scratch is a time consuming and complicated process. For that reason, we opted to use models that were pre-trained on either the PASCAL Visual Object Class (“The PASCAL Visual Object Classes Homepage.” 2012.) or the Common Objects in Context image set (“COCO - Common Objects in Context” n.d.). Although neither of these datasets contain chicken objects, the initial layers in the neural network, initialized to the weights determined by the training on these image sets, serve significantly better than any other initialization technique.

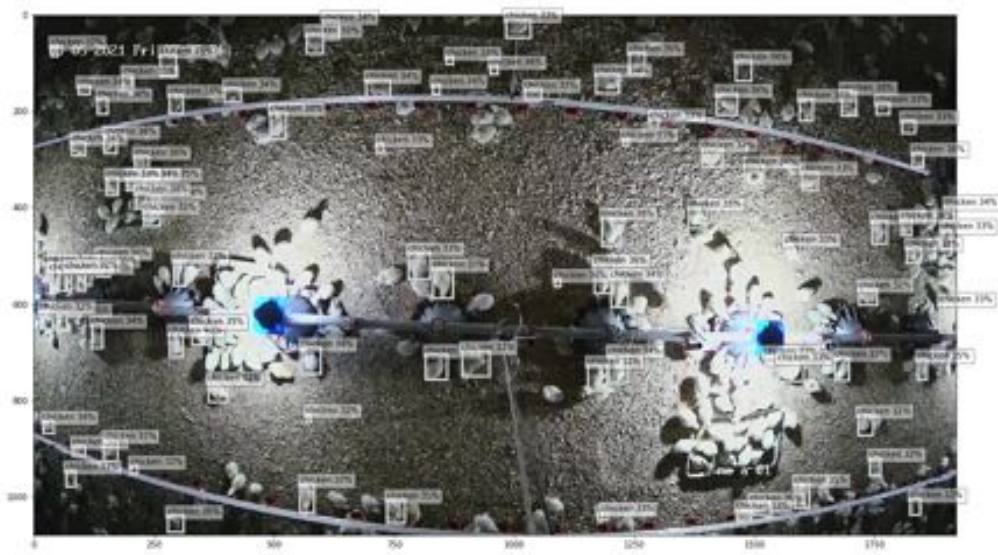


Figure 7: *jumpstart-ftc-mx-od-ssd-512-vgg16-atrous-coco-adreo*: results of using the COCO pre-training image set, and vgg feature extraction model



Figure 8: *jumpstart-fic-mx-od-ssd-512-vgg16-atrous-voc-adreo_post_filter*:
results of using the **VOC** image set

The COCO and PascalVOC data sets detected a comparable 95 and 96 instances of chicken respectively. Despite this similar number of detected instances, the COCO model seems to detect more chickens, as the PascalVOC model seems to have a high degree of overlap between the detected chicken objects. The PascalVOC model also seems to perform much better in the low light areas, i.e., the corners of the images, as can be seen when comparing figure 7 and 8. This seems to indicate the model trained on PascalVOC is the superior training set.

Image downsampling size

Almost all neural networks use downsampling as one of the first layers to eliminate the requirement for input images being a specific size and to reduce training time and model complexity. MxNet offers two different resolutions, 300 (example results in figure 9) and 512 (example results in figure 7). These correspond to the training image size supported. For example, the 300 corresponds to a down-sampled resolution of 300x300 pixel image. The 512 resolution model applied to our test image can be seen above in figure 7.

The 300 model required all training images to be downsampled to a smaller image size as compared to the 512 model. Due to the relatively small size of chickens compared to the image as a whole, we expected the additional resolution available in the 512 model to outperform the 300 model. This was not the case however, as we saw roughly 9% more false positives in the 512 model, with an example shown in figure 10. These results seem to indicate our 300-coco model outperformed our 512-coco model.

bounding box may correspond to one of many ground truth chickens. Potential solutions to remedy this include non-maximal bounding box suppression. (N. O. Salscheider), however this would force labeling of groups of chickens as a singular chicken entity.

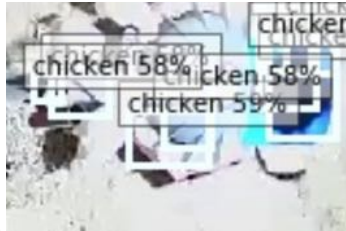


Figure 11: Example of cluster of chickens being labeled as one object

Intersection over Union (IoU)

Intersection over union (IoU) represents another performance metric that is specifically suited for object detection. It is calculated by dividing area of overlap over area of union. Both of these areas are calculated using the ground truth and predicted object bounding boxes respectively. Due to the model predictions being limited to 100 object instances, and the problem of matching predicted to ground truth bounding boxes, this metric was infeasible to calculate accurately.

Loss Metrics

There are two types of loss metrics calculated for these models evaluated: L1 loss and cross entropy. L1 loss is calculated as the mean absolute error between the label and predicted value. Cross entropy, or log-loss, is a measure of probability of the predicted object being close to one for true positives. Another way of saying this is that cross entropy grows large as the probability of predicted classes approaches zero.

Selected model

After comparing the various models and characteristics, we selected the 300x300 vgg16-atrous model pre-trained on the COCO image set based on the minimized loss.

2.5 Model Creation Process

All the models were built within AWS Sagemaker Studio using Jumpstart, a tool containing a collection of different ML model frameworks that can be used for object detection, regression, text classification, and more. The object detection

models are pre-trained and already included classes (e.g., people, cars). Chickens are not included in these classes, so we had to implement fine-tune training to build the object tracking model on our custom dataset of labeled chicken images. Fine-tuning on top of a pre-trained model allows for better results with fewer labeled images.

Only a small number of pre-trained models could be used for fine-tune training in Sagemaker, and we built our model on top of the SSD VGG16 Atrous 300 model from MXNet.

After training, for any given model we built, we were able to deploy an endpoint in Sagemaker that we could use for performing object detection.

3 OUTPUT ANALYSIS

3.1 Predicted Class Filtering

As an output of several of our models, we saw instances of large sections of the input image being classified as a chicken object. This could be due to non-maximal suppression being improperly applied.

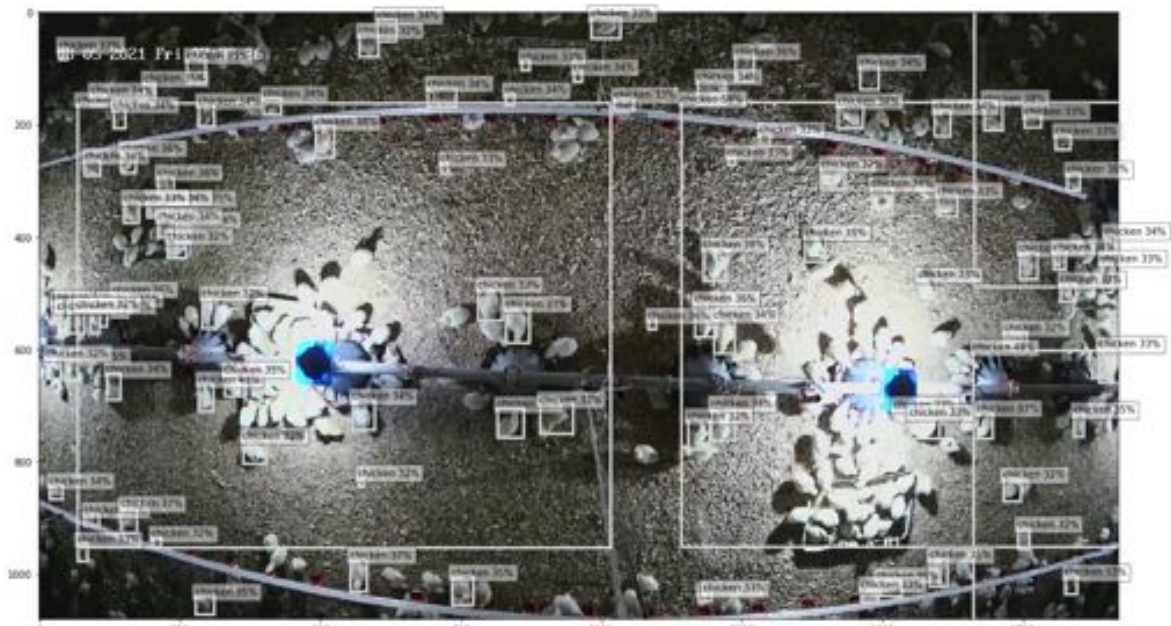


Figure 12: *jumpstart-ftc-mx-od-ssd-512-vgg16-atrous-coco-adreo_pre_filter.png*

To resolve these obviously false results, we applied an area filter to remove the chicken objects that were above a certain area threshold.

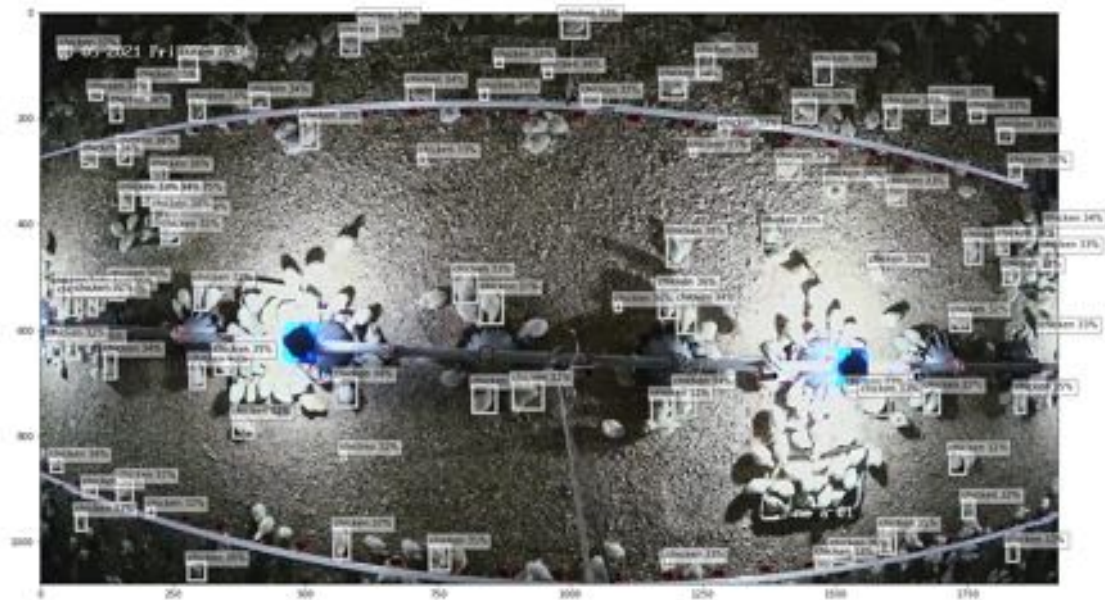


Figure 13: *jumpstart-ffc-mx-od-ssd-512-vgg16-atrous-voc-adreo_post_filter.png*

As can be seen, this successfully removes all instances of these oversized false positives programmatically without removing any true positives.

3.2 Model Tuning

Using the pretrained models within AWS Sagemaker's Jumpstart framework limited the amount of model tuning required. The first hyperparameter we tuned was the number of epochs, or training iterations. Normally both validation and training loss would be measured separately on different training sets, resulting in a true optimization between over and under training the model, however this was not possible given the ML framework.

Instead, we chose to handle this optimization using loss and cross entropy. From the below entropy-loss curve in figure 14, we can see that both these metrics are minimized at roughly 60 epochs of training. After that we see diminishing returns, which risks overfitting the model.

The next hyperparameter investigated was the learning rate. While the higher learning rate of 0.05 caused the loss to converge to steady state values in fewer epochs, it also resulted in increased fluctuations in both performance metrics. These results can be compared below in figures 14 and 15. Since both steady-state values were roughly identical, we opted for the lower learning rate to decrease steady state fluctuations. A future study on this could consider using dynamic learning rates that start high and end low to achieve the best of both options.

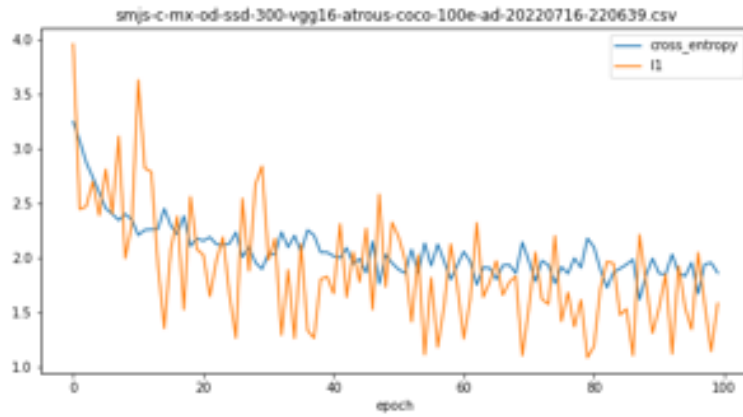


Figure 14: L1 loss cross entropy using a learning rate: 0.01 (default)

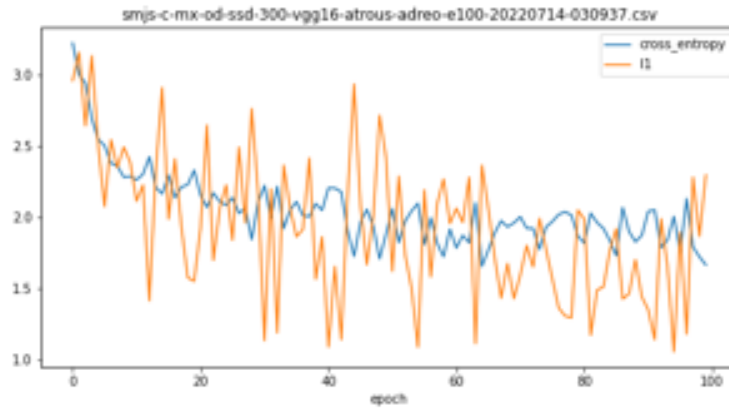


Figure 15: L1 loss cross entropy using a learning rate: 0.05

3.3 Model Shortcomings

While going through the labeling and model building process we were able to identify a few areas that limited the ability of the models.

Overexposure

One issue that is clear from looking at the training videos is the lack of contrast between the chickens and the ground near the light sources. As a result, we ended up with groups of chickens that were severely overexposed. In areas like this, it was often impossible to clearly identify chickens even for humans, meaning a model couldn't be expected to perform this task either.



Figure 16: Overexposed image

Underexposed and Distorted regions

Another issue was underexposure and distortion in the video, where chickens could not be detected from the black background. This was particularly apparent in the corners of the videos, as shown below. Although these are still individual chickens, they look different than the chickens that are more centered in the field of view.



Figure 17: Underexposed images

Tight groupings of chickens

There were also several instances of highly clustered chickens, where individual chickens could not be easily differentiated from the cluster as a whole. Rather than risking the model identifying clusters of chicken, we left these chicken objects unlabeled.

Labeling Differences

The aforementioned challenges (lighting, image distortion in corners of the frame, tight grouping, etc.) also led to differences in labeling. Although we discussed beforehand and decided to only label chickens that we thought could be identified, there was still subjectiveness in the labels of chickens between team

members. A few examples of the types of labeling differences we saw are shown below.

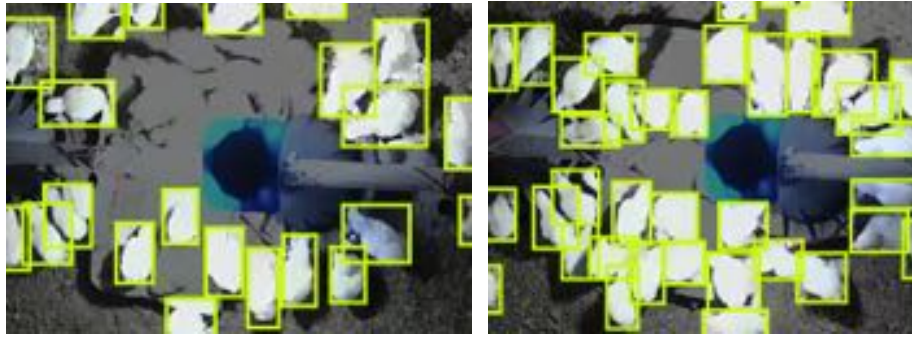


Figure 18: Labeling differences with overexposure from ground lighting.

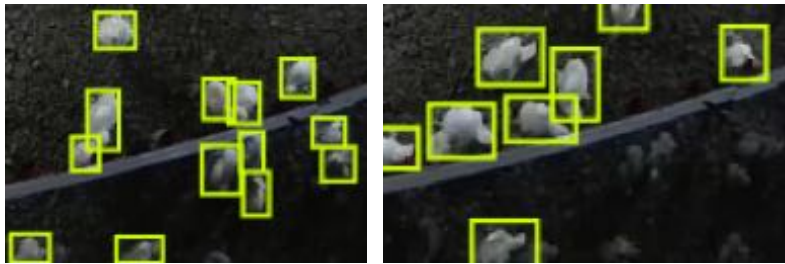


Figure 19: Labeling differences in the corner of the video frame.

4 MODEL APPLICATION

Using the object detection models, we implemented an object tracking algorithm and heatmap tool. The object tracking algorithm detects and tracks chicken locations, gathering movement data and creating new videos that show the trackers. The heatmap uses the predicted locations and visualizes them into the same frame dimensions. The use of these tools is to provide the poultry farms with the insights to improve animal welfare.

4.1 SORT Object Tracking

SORT Background

Simple, Online, and Realtime Tracking (SORT) object tracking is a tracking algorithm used for multiple object tracking (MOT) and is “based on rudimentary data association and state estimation techniques” (Bewley, et al., 2016).

This algorithm “doesn't handle occlusion or re-entering objects,” but was “ranked the best open source multiple object tracker on the MOT benchmark,” when the first paper describing the implementation was published in 2016 (abewley's SORT repository on Github).

The trained model is “a key factor influencing tracking performance,” so shortcomings in object tracking with SORT may be due to the object detection model used (Bewley, et al., 2016).

SORT Implementation

Using our trained chicken object detection model and the SORT algorithm, we were able to learn from a YouTube video (“Easy Object Tracking with a Kalman Filter in OpenCV” by Nicolai Nielsen) and its associated implementation in Github and modify it with our own code to update it for our purposes. This included integrating with S3 buckets and the endpoint from our custom, trained model to predict object detections, updating or rewriting certain functions like drawing bounding boxes, creating the output video, and capturing and analyzing movement data from a video clip.

We manually found several clips that exhibited different levels of mass movement or minimal movement to compare the tracking results and determine if average speed could be used as a metric to identify mass movement events. The first clip which exhibited mass movement (taken from the ch03_20210305073839.mp4 video file) was selected to show a movement event of average size while the second mass movement clip (taken from the ch03_20210305085439.mp4 video file) was selected to show a very large movement event.

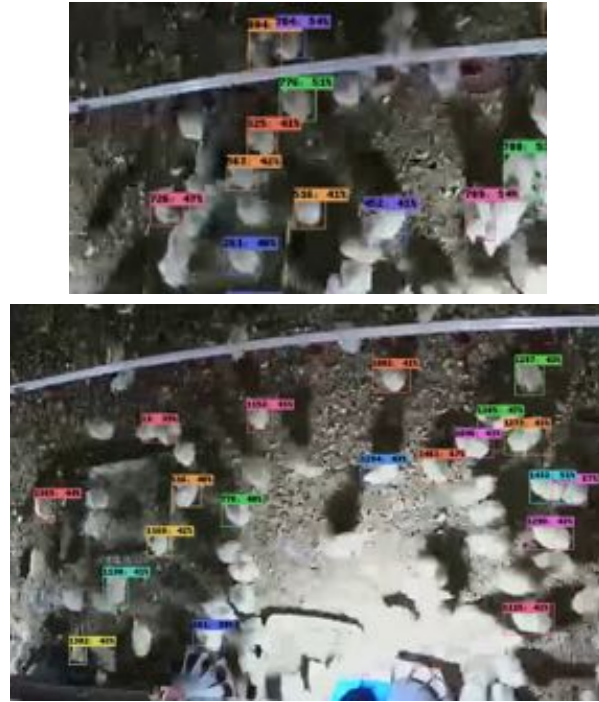


Figure 20: Tracking during mass movement



Figure 21: Tracking with minimal movement

As shown, the combined object detection model with object tracking was able to accurately identify and track chickens. Using these trackers, we were able to extend our analysis to identify average speed and understand whether clips exhibited movement events.

Centroids were calculated for all bounding boxes and the number of frames that a tracker existed was captured. This allowed for the calculation of the distance an individual chicken moved from the time its tracker was instantiated to the time it was destroyed.

Using this information, we calculated average pixels that the chickens moved per frame and per second. All video clips had the same frame rate (30 fps), but if comparing clips with different frame rates, the average distance moved (in pixels) per second would be a better metric. Given a known distance measurement in the video's field of view we could have translated the pixels moved into an average speed, but this information was not available. This might also require a mapping to account for any distortion or fish-eye effects from the camera.

SORT Results

Results are shown below for predictions made with two different trained models. Values for the number of pixels moved were found by averaging over all chicken trackers in the video clip.

Video Clip	Average pixels moved (per frame)	Average pixels moved (per second)	Note
ch03_20210305073839_short1move.mp4 (clip 1 move)	1.14	34.20	Mass movement
ch03_20210305073839_short1no.mp4 (clip 1 no)	0.34	10.11	Minimal Movement
ch03_20210305085439_short2move.mp4 (clip 2 move)	3.10	92.87	Lots of Mass Movement
ch03_20210305085439_short2no.mp4 (clip 2 no)	0.44	13.21	Minimal Movement

Table 1: Using trained model built with “SSD VGG16 Atrous 300” (training job name: “mx-od-ssd-300-vgg16-atrous-coco”, learning rate: 0.01, batch size: 4, epochs: 50) and its associated endpoint.

Video Clip	Average pixels moved (per frame)	Average pixels moved (per second)	Note
ch03_20210305073839_short1move.mp4 (clip 1 move)	0.91	27.17	Mass movement
ch03_20210305073839_short1no.mp4 (clip 1 no)	0.37	10.99	Minimal Movement
ch03_20210305085439_short2move.mp4 (clip 2 move)	2.26	67.69	Lots of Mass Movement
ch03_20210305085439_short2no.mp4 (clip 2 no)	0.46	13.67	Minimal Movement

Table 2: Using a trained model built with “SSD VGG16 Atrous 300” (training job name: “smjs-c-mx-od-ssd-300-vgg16-atrous-coco-100e-ad-20220716-220639”, learning rate: 0.01, batch size: 4, epochs: 100) and its associated endpoint.

It is clear that both models are able to differentiate between a clip with minimal movement and a clip with mass movement. There is also a clear difference between mass movement events of different sizes.

It is also informative to look at histograms of the average movement per second for all trackers found in each video clip. I focus on the second model for this portion of the analysis.

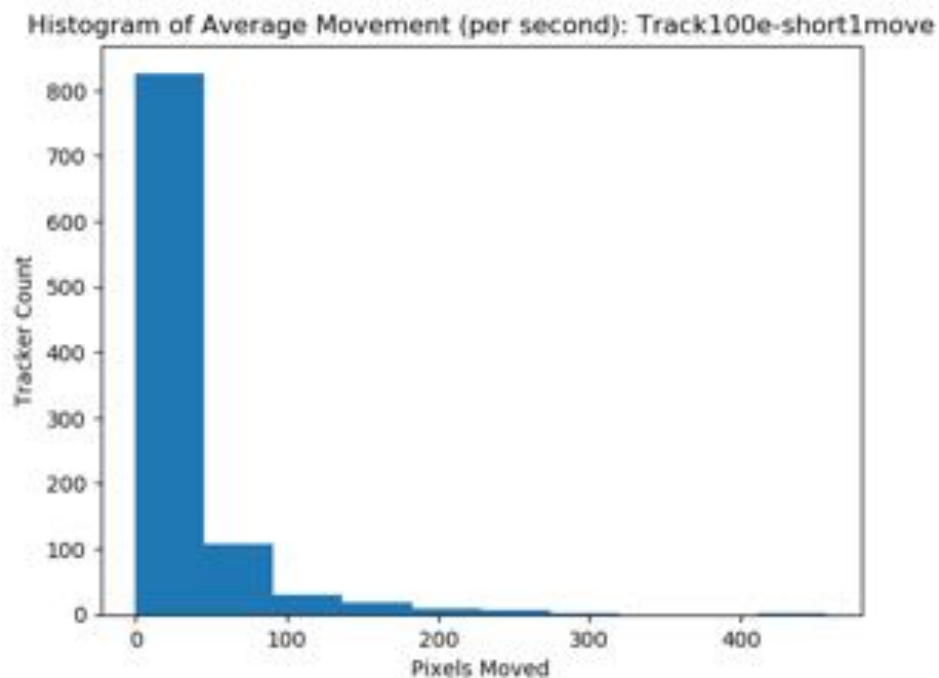


Figure 22: Average movement per second (clip 1 move)

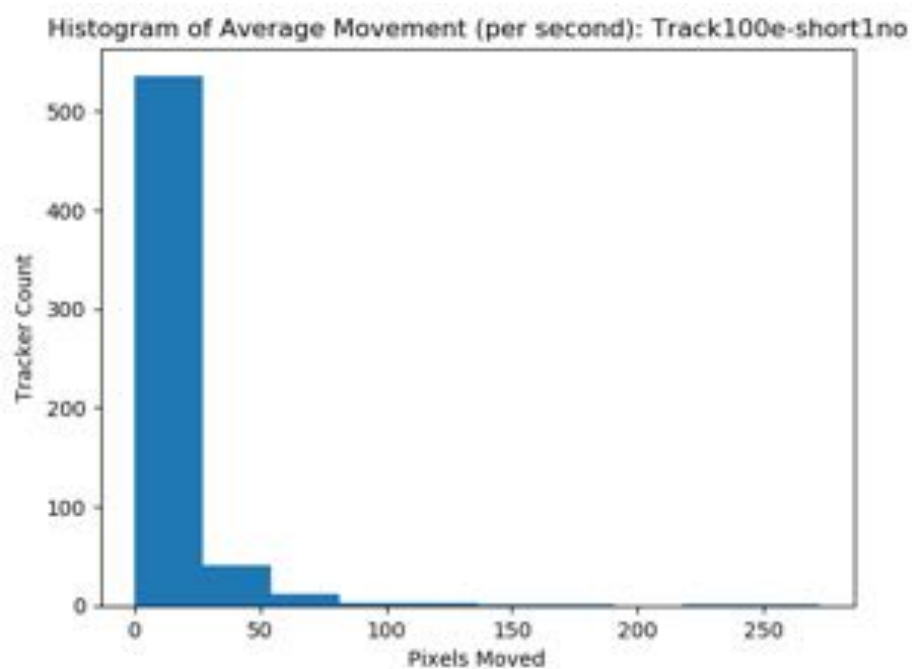


Figure 23: Average movement per second (clip 1 no)

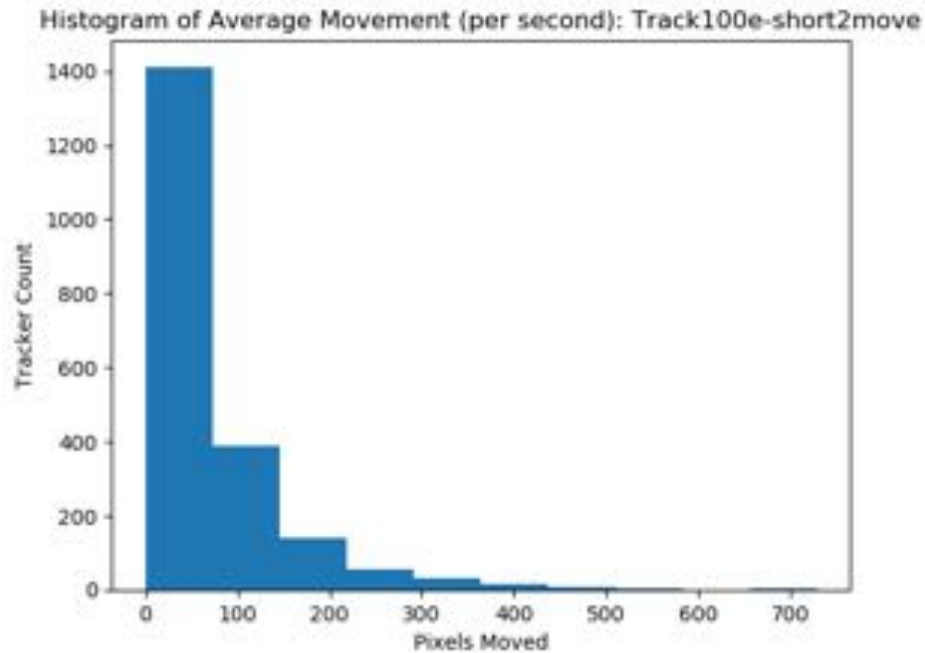


Figure 24: Average movement per second (clip 2 move)

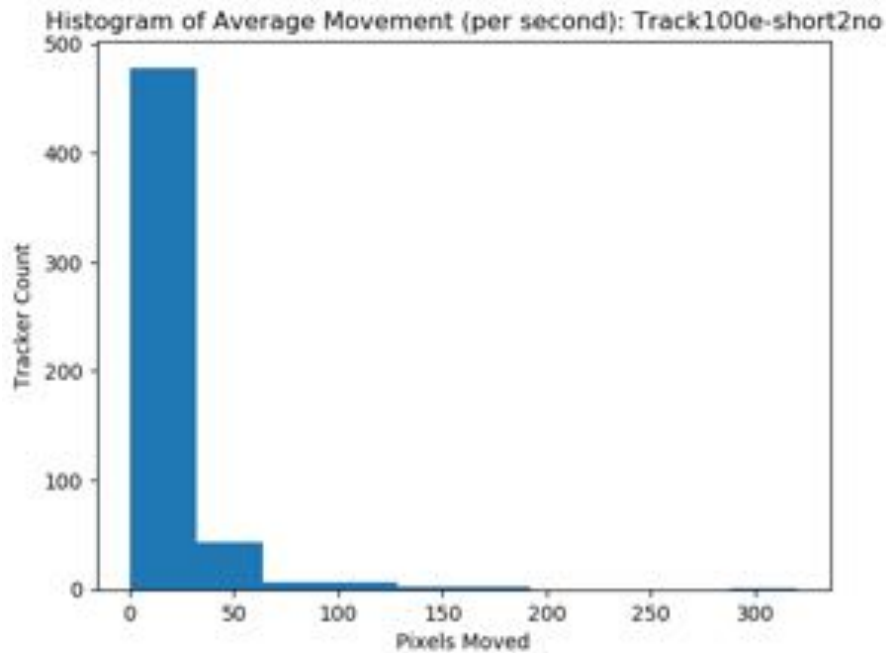


Figure 25: Average movement per second (clip 2 no)

While large numbers of objects that displayed small amounts of movement were detected and tracked in each video clip, the histograms of mass movement events have heavier tails and indicate there are more chickens that move farther distances.

One reason for the large number of trackers that displayed very little movement (even during mass movement events) appears to be due to background/ground cover being detected as chickens. By watching the resultant videos with tracking boxes, it becomes clear that these false positives are being detected around the edges of shadows in a small section of the field of view. The wood chips on the ground look similar to many of the chickens labeled in the shadowed and distorted corners of the training dataset, and are being detected as chickens where the harsh lighting conditions allow for it. They are further being identified as tracked objects since the change in lighting (due to the movement of a chicken's shadow) leads the algorithm to believe that they are moving.

These trackers don't last long, but the false positives aren't moving any appreciable distance so they artificially decrease the average calculated speed and the histograms end up with more trackers that show minimal movement than we should actually have.

Although we varied SORT parameters to visually try and determine the best values, adjusting them further might be able to help minimize this effect. We could also filter out these detections since they are generally smaller than other nearby chickens. A location-dependent filter would have to be used since small detections might be correct in the corners of the video.

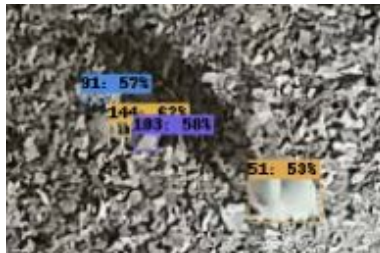


Figure 26: False positives around a chicken's shadow



Figure 27: Numerous false positives around shadows during a mass movement event

Even though we encountered this issue with false positives in one section of the video, this did not affect our results and our object tracking implementation was able to track movement and can be used to identify mass movement events reliably.

4.2 Heatmap

The heatmap is created from the coordinates of the bounding boxes to illustrate the positioning of the chickens. The visual provides the magnitude of the chickens throughout the defined sections of the house.



Figure 28: Heatmap of identified chickens

The value of this tool comes from using a collection of outputs from the model to calculate the average chicken population in each section. This information informs the farmers of chicken movement activity over the course of a set time period. Knowing the average density of the chickens throughout the course of an hour or day could alert the farmer to take a specific action.

For example:

- A higher density indicates more cleaning as chickens are producing waste more frequently in these areas.
- An average density close to zero may contain some revolting substance that the animals avoid (a dead rodent or a chemical spill)
- Specific areas can be monitored to provide insight into chicken activities
 - Times that a lot of chickens are drinking or eating
 - Are chickens going to the food and water stations
 - Movement patterns over the course of the day
 - This would require comparisons between heatmaps

Heatmap Tool Details

The tool functions by finding the center of the bounding boxes for each of the identified birds. Then for each of the sections defined a summation of the total number of birds is calculated. This number is used to determine the ‘heat’ of the segment with the color coming from a color palette, named `rocker_r`, found in the `seaborn` library (Waskom).



Figure 29: Complete color palette

As we wanted the tool to show increasing redness for the heatmap, only the left half of the palette was used to select colors for the heatmap.



Figure 30: Portion of color palette used for heatmap

The tool allows for customization of the number of heatmap sections to divide the poultry house into, providing flexibility in how to monitor the chickens and allowing for the number of chickens identified in each section to be made visible in the output if desired.



Figure 31: Heatmap with 25 sections

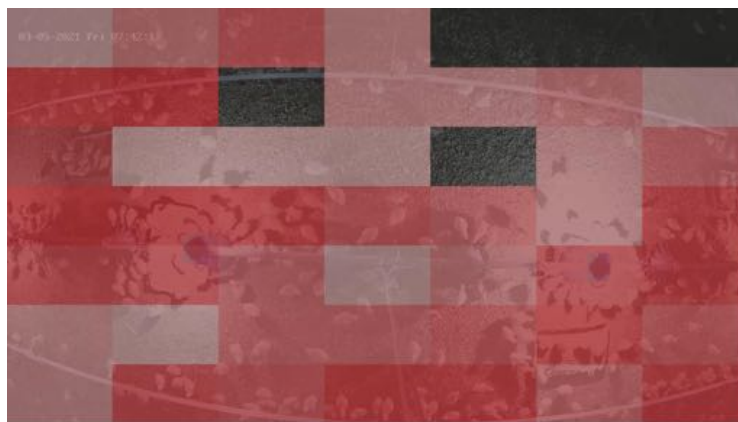


Figure 32: Heatmap with 49 sections



Figure 33: Heatmap with 144 sections

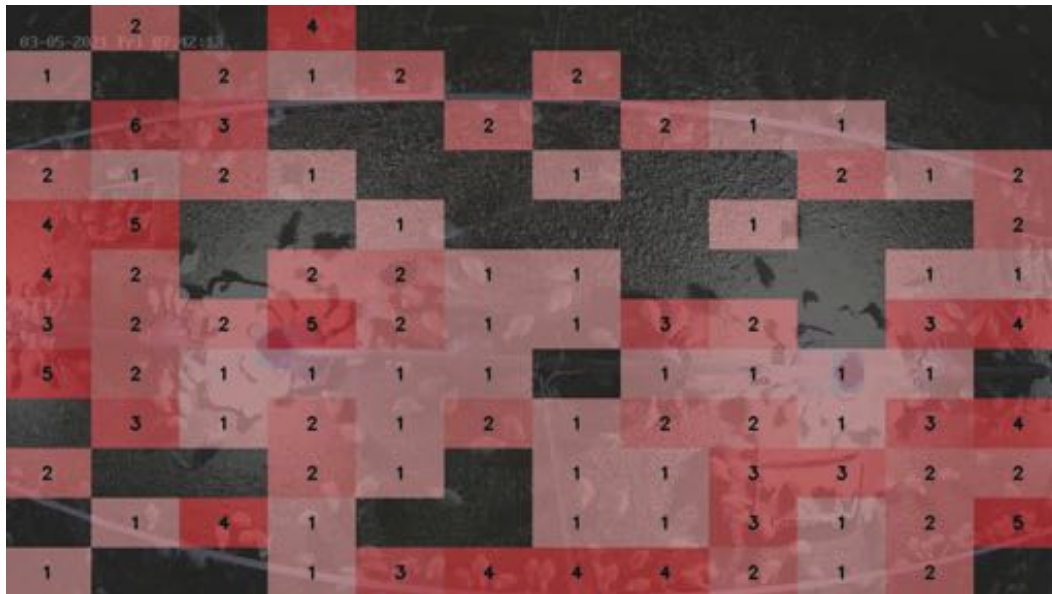


Figure 34: Heatmap with 144 sections showing number of chickens per section

Considerations

It is important to note the segments of the heatmap do not represent the same physical dimensions of the poultry house floor. This is due to the distortion or fish-eye effects from the camera. The sections of floor on the sides of the image cover a larger area of floor; therefore, by the simple constraint of floor space, the center sections have a lower limit for maximum capacity.

5 CHALLENGES

Aside from the model shortcomings detailed previously, there were external challenges encountered over the course of this project.

A major issue faced was working within the Amazon Web Services (AWS) suite of products. Before progress could be made, we first needed to learn how to operate within the AWS environment. This requires files and folders to follow a very specific format in order to be accessible for the models. Additionally, the AWS platform has a complex access management structure that brought about permissions issues over the course of the project.

After resolving the above issues, the next challenge was actually learning to use the tools within AWS. With no prior experience with Sagemaker and Ground Truth, there was a heavy learning curve before reaching a state where we had a functional pipeline to start labeling data, processing that data, and then building and deploying trained models.

The next and biggest challenge we encountered was a limitation with the classifying capabilities within the AWS Sagemaker models. Despite the labeled images having on average around 280 chickens per frame, the models were only able to classify a maximum of 100 items per image. This severely limited analysis of the number of chickens moving around the poultry house.

The final challenge was inaccessibility to tools outside of AWS that are commonly used in concurrence with Sagemaker and Ground Truth, specifically Roboflow, which was referenced in several of the online learning resources for AWS Ground Truth. The platform automatically converts Ground Truth labeling jobs into a usable annotation format (e.g., COCO, PascalVOC) and augments labeled images to generate a larger training and testing set. However, given the proprietary nature of the project, we were not able to leverage the free version of the platform.

6 CONCLUSION

From previous analysis, we determined that the 300x300, vgg16-atrous model pre-trained on the COCO image set performed better than the several other base model combinations considered. We further found that the optimal hyperparameters of 60 epochs and a learning rate of 0.01, led to optimal results.

Using our object detection models, we built an object tracker with the SORT algorithm that automatically labels chickens in video clips. We wrote ancillary

code that was able to capture movement data, and using these results we were able to accurately detect mass movement events.

Our object tracking model could be used with an appropriate threshold to improve animal welfare by notifying a farmer that a mass movement or other anomalous behavior has occurred.

7 NEXT STEPS

There are several future actions that could be taken to build upon the results and enhance our analysis.

7.1 Normalize Brightness and Distortion

Many labeling, training and modeling flaws were caused by lighting and distortion issues. Solutions such as post-processing or cropping of the videos should be explored. These techniques could make the objects being labeled more obvious, thereby increasing labeling consistency.

7.2 Label Additional Types of Objects or Behaviors

The object tracking model we built had trouble detecting and staying with birds moving quickly in a mass movement event. This may be due to the fact that these types of events are infrequent and our labeled dataset did not include these types of events. Although this would probably be less critical for an object that doesn't change shape at different speeds (like a car), fast moving chickens can change shape somewhat from frame to frame as they flap their wings or as they appear blurred in the video capture. Labeling mass movement frames could improve the performance. Object detection for chickens that look different during different activities (besides mass movement) may also struggle if not explicitly labeled.



Figure 35: Moving chickens

7.3 Increase The Video Frame rate

The object tracking model we built was able to detect chickens in motion, but it appeared less capable for faster moving chickens especially as they moved towards the side of the video (and sometimes into overexposed lighting conditions at the same time). Although we tried to decrease the IOU parameter in the


algorithm to allow for better tracking, it appears that fast moving chickens may move quickly enough that sometimes there is minimal overlap from frame to frame.

7.4 Augment the Training Data

We labeled over 6000 chickens in 21 video frames from a single video (ch03_20210305073839.mp4). This equates to over 280 chickens per frame. Although this seems like a large number of labeled chickens, some chickens appear vastly different from others when in the far corners of the frame or under harsh lighting. Adding more labeled images (including different behaviors) may help.

The dataset could also be augmented by randomly adjusting different image settings and saving those labeled files as additional training images. For example, we could randomly adjust a labeled image's brightness or contrast up or down or add noise to increase the size of the dataset without having to label additional images.

8 REFERENCES

1. 
2. Foundation for Food & Agriculture Research. "Foundation for Food & Agriculture Research," May 25, 2022. <https://foundationfar.org/>.
3. NCCAdmin. "National Chicken Council | U.S. Broiler Performance." National Chicken Council, February 15, 2022. <https://www.nationalchickencouncil.org/about-the-industry/statistics/u-s-broiler-performance/>.
4. National Chicken Council (NCC). "National Chicken Council | 2020 U.S. Broiler Chicken Industry Sustainability Report," September 29, 2021. <https://www.nationalchickencouncil.org/industry/sustainabilityreport/>.
5. "United States Department of Agriculture Poultry -Production and Value 2020 Summary," 2021. https://www.nass.usda.gov/Publications/Todays_Reports/reports/plva0421.pdf.
6. Amazon Web Services. "Streamlining Data Labeling for YOLO Object Detection in Amazon SageMaker Ground Truth | Amazon Web Services," October 14, 2020. <https://aws.amazon.com/blogs/machine-learning/streamlining-data-labeling-for-yolo-object-detection-in-amazon-sagemaker-ground-truth/>.
7. Bewley, Alex, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. "Simple Online and Realtime Tracking." 2016 IEEE International Conference on Image Processing (ICIP), September 2016. <https://doi.org/10.1109/icip.2016.7533003>.
8. abewley. "Abewley/Sort: Simple, Online, and Realtime Tracking of Multiple Objects in a Video Sequence." GitHub, November 28, 2020. <https://github.com/abewley/sort>.

9. niconielsen32. "Niconielsen32/ObjectTracking." GitHub, July 2, 2022.
<https://github.com/niconielsen32/ObjectTracking>.
10. "Easy Object Tracking with a Kalman Filter in OpenCV." YouTube Video.
YouTube, July 4, 2022. <https://www.youtube.com/watch?v=oY4DXtZlZ-k>.
11. Waskom, Michael. "Choosing Color Palettes." seaborn. Accessed July 19, 2022.
https://seaborn.pydata.org/tutorial/color_palettes.html.
12. "Detection — Gluoncv 0.11.0 Documentation." Cv.gluon.ai,
cv.gluon.ai/model_zoo/detection.html#ssd. Accessed 19 July 2022.
13. "The PASCAL Visual Object Classes Homepage." *Ox.ac.uk*, 2012,
host.robots.ox.ac.uk/pascal/VOC/.
14. "COCO - Common Objects in Context." *Cocodataset.org*,
cocodataset.org/#home.
15. N. O. Salscheider, "FeatureNMS: Non-Maximum Suppression by Learning
Feature Embeddings," 2020 25th International Conference on Pattern
Recognition (ICPR), 2021, pp. 7848-7854, doi:
10.1109/ICPR48806.2021.9412930.